DOCUMENT RESUME

ED 466 588 IR 021 323

AUTHOR	Hall, Jon; Rapanotti, Lucia
TITLE	Educational JavaBeans: a Requirements Driven Architecture.
PUB DATE	2001-00-00
NOTE	7p.; In WebNet 2001: World Conference on the WWW and Internet Proceedings (Orlando, FL, October 23-27, 2001); see IR 021 310. Some charts contain small, blurred text.
PUB TYPE	Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE	EDRS Price MF01/PC01 Plus Postage.
DESCRIPTORS	Case Studies; Computer Software; *Computer Software Development; *Computer Uses in Education; *Distance Education; Foreign Countries; Higher Education; *Interaction; Models
IDENTIFIERS	*Electronic Commerce; Open University (Great Britain)

ABSTRACT

This paper investigates, through a case study, the development of a software architecture that is compatible with a system's high-level requirements. The case study is an example of an extended customer/supplier relationship (post-point of sale support) involved in e-universities and is representative of a class of enterprise without current architectural support. The relevant high-level business processes are described. Requirements and challenges of the current business model are highlighted, and discharging and meeting them through electronic support is explored. Specifically, the paper considers the current manifestation of the distance learning educational dialog within the Open University (United Kingdom), including student/student recruitment team interaction (receiving materials), student/tutor interaction (studying), and tutor/student interaction (supporting a student's learning). The Enterprise JavaBeans (EJB) blueprint architecture is proposed, and its compatibility with these requirements and challenges is argued. Finally, the paper discusses the applicability of the new blueprint architecture in a wider enterprise setting. (Contains 14 references.) (MES)

Educational JavaBeans: a Requirements Driven Architecture

Jon Hall and Lucia Rapanotti
Computing Department
The Open University
Walton Hall
Milton Keynes, UK
{J.G.Hall, L.Rapanotti}@open.ac.uk

Abstract In this paper we investigate, through a case study, the development of a software architecture that is compatible with a system's high-level requirements. The case study is an example of an extended customer/supplier relationship (post-point of sale support) involved in e-Universities, this being representative of a class of enterprise without current architectural support.

In detail, we describe the relevant high-level business processes. We highlight requirements and challenges of the current business model and explore discharging and meeting them through electronic support. We propose a new blueprint architecture and argue that it is compatible with these requirements and challenges. Finally, we discuss the wider applicability of the new blueprint architecture in a wider enterprise setting.

1. Introduction

With increasing demands being made for the electronic support of distance learning and teaching, there are many reasons why an architecture for *electronic Learning and Teaching* (eLT) is important and timely:

- initiatives such as the EU Ariadne project (http://ariadne.unil.ch/) or the UK e-University (http://www.hefce.ac.uk/Pubs/default.asp) introduce new challenges for the Educational Engineer to include infrastructural electronic support for the business of learning and teaching at a distance;
- platform independent technologies are available which already go some of the way towards supporting the business model.

In fact, educational software development has never been so central to the success of a university. The UK education market is not without good electronic support; the Open University being only one of many institutions making use of information technologies in their teaching. Current development practice can, however, only be described as *ad hoc*. One of the reasons for this, we feel, is the lack of a proper foundation for the building of educational software systems. We have not yet reached a 'technological critical mass' for educational software development.

On the other hand, such a critical mass has been reached for eCommerce systems, witness the recent explosion of electronic commerce products. Technologies such as the Java 2 Enterprise Edition (J2EE) platform (Shannon et al. 2000), which includes the Enterprise JavaBeans™ (EJB) architecture, have provided the catalyst for eCommerce systems development. In this paper, we hope to go someway towards identifying the catalyst for education software development. We wish to build on the firm bases provided by EJB to extend its properties and philosophy to cover distance learning and teaching.

1.1 eLT Products, Services and Business Processes

There are strong similarities between many of the business processes of commerce operations and those within distance learning and teaching. In the most general case, both have customers; both work on a contract basis; both provide end-user support; both work through intermediaries. Moving into the electronic setting of eCommerce and eLT, we may deduce that EJB goes someway towards servicing the electronic support needs of eLT. However, with few exceptions, eCommerce has concentrated on the exchange of physical goods for which post-point of sale electronic support is unnecessary - delivery and use happen in the material rather than the electronic world. In eLT, the commodity on offer is information employed by the end-user (potentially) every day; in this case, continued close involvement in the product is integral with its delivery and use.

Supporting (and extending) the richness of the customer/supplier dialog whilst building on previous best practice provides our motivational focus for the development of the blueprint architecture proposed in this paper, the Educational JavaBeans (EdJB) architecture. In the next section we focus on eliciting requirements for the support of the distance learning educational dialog. We begin by considering its current manifestation within the Open University, the major provider of distance education in the UK. As this is not a paper on the distance learning model *per se*, for brevity we stop short of describing the minutiae of the model, concentrating on the relevant higher-level aspects of the model.

2. Supporting Current Practice in Distance Learning

The Open University (OU) teaches, at a distance, to over 150,000 students per year. The OU teaching model - also called *supported open distance learning* - has evolved over thirty years to its current efficient and effective form. Because their use is best understood, it bases teaching on text technologies, i.e., books, electronic documents, web pages, continuous assessment and exam papers, but there is some *ad hoc* use of supporting media - such as television, audio cassettes and computer-based instruction.

The business processes underlying the OU teaching model are well defined, effective, efficient and scaleable, and so provide a rich source of dialogs between stakeholders for requirements elicitation. For the purpose of our discussion, we identify the following stakeholder groups within the OU model: *students*, *associate lecturers* (ALs, also called *tutors*), *course authoring teams*, and *student recruitment teams*.

In this section we consider, from a high-level perspective, dialogs between those stakeholders. Each dialog description is structured as follows: we describe the dialog as it exists and from this description determine minimum requirements for the EdJB architecture to be able to provide support. We then identify the 'challenges' that exist within the current model that we feel can be met by 'extended' electronic support.

Reality check: So that this paper is grounded in reality, we must consider the quality of electronic support that is currently available and sustainably available in the future. We will therefore not assume that electronic support is required as a 24/7 thing (i.e., 24 hours per day, 7 days per week), only that there will be relatively short, non-contiguous periods with the end-user in electronic contact with the supplier. Moreover, neither will we assume that the supplier can determine nor influence *a priori* the time that the end-user spends on-line. The justification for this mode of operation comes down to the fact that it is the customer who determines product use - even when that product is electronically supported. This is clearly unlikely to change.

The following section describes the dialogs that motivate the architecture. A dialog description is followed by derived requirements (labeled **R**<requirement number>, for reference). Then, in a box, we identify challenges, i.e., things that cannot yet be done, but with electronic support may be possible. Such challenges again source requirements, labeled as above.

2.1. Student/Student Recruitment Team Interaction: Receiving Materials

Currently, the primary delivery mode for learning materials is by paper mail at the start of the course. When necessary, a second major mailing will provide further material later on during the course. Minor mailings will then be used to inform students of errata in the course materials, should errata be brought to the course authoring teams attention during a course presentation.

Derived requirements – The architecture must:

R1 provide support for the publication and delivery of teaching materials

Challenge

It is important to note that, due to the present reliance on paper-like media, having the characteristic of being unchangeable when 'published', course texts remain unchanged, often for many years, even when problems with the material are known[1]. Even though every effort is made to inform students of errata, a student will often be

[1] Another way of seeing these characteristics is that the OU is tied to the use of media that require long 'print' runs for them to be economic. Even certain electronic media, such as CDROMs, share such characteristics.

unaware of the presence of errata, or become aware of them a long time after the error was encountered. This can have a serious and deleterious effect on the learning and teaching experience, which can contribute greatly to student (and so customer) dissatisfaction. Providing a changeable (i.e., electronic, server-side) media base can address this challenge, as long as suitable materials management, such as versioning and configuration management technologies, are also available.
Derived requirements – The architecture should:

R2 provide support for the use of changeable teaching materials as the basis of the teaching provision

2.2. Student/Tutor Interaction: Studying

Study is deadline-driven by continuous assessment components (CACs). Tutor marking of CACs provides validation and feedback on a student's learning experience. For a half credit course (one academic year's part-time study) there will be between 6 and 8 CACs; tutor marking of assessment material - closing the feedback loop between student and university - can therefore typically occur as infrequently as 5 to 7 weeks.
In addition, synchronous interactions are widely acknowledged by students and teachers alike as being very valuable. Until recently, the only means of achieving them was by telephone - with the obvious restriction on number of participants - or by personal contact - which requires travel to and, for some, an unwelcome commitment of time at some study centre. Currently, some electronic support is available, using conferencing tools such as Lyceum (Rapanotti & Hall 2000), and their use should be facilitated by the architecture.
Derived requirements –The architecture must provide support for:

R3 periodic external study validation

R4 for synchronous interactions at a distance

Challenge
Between CACs, *self*-assessment materials (SAMs) provide students with feedback on much shorter time scales– typically on a *per* subsection study basis (i.e., from minutes to hours) - answers being provided along with the teaching text. Although often enriching, such non-external validation mechanisms can also impact negatively the student experience: although success can breed success, failure on SAMs can lead to student demoralisation. Reasons for failure are twofold: the student has not understood the materials on which they are assessing themselves; there are errors in the SAMs (which unfortunately can happen). Given that external validation can be 5 to 7 weeks away, this can lead to students becoming seriously misguided before any corrective action can be taken. Currently, there is no university-wide mechanism for tutor-validation (or otherwise) of activities between CACs. What is needed is a dialog between student and tutor that has a much finer granularity than the current 5-7 weeks but which overloads neither the student nor the tutor.
Derived requirements – The architecture should:

R5 provide support for periodic external study validation at a fine granularity (i.e., *per* SAM)

2.3. Tutor/Student Interaction: Supporting a Student's Learning

Tutors are the embodiment of the university during course presentation: they provide support for learning and pastoral care to students. Tutors mark their students' scripts. They are assisted in this task by 'tutor notes' issued by course authoring teams and including solutions and a marking scheme. Tutors receive the scripts from their students at particular deadlines set by the course study calendar, and return the marked scripts to the OU headquarters for processing – registering marks and having their marking monitored – from where they are returned to the students. Script submission, marking and processing are largely text-based.
Derived requirements – The architecture must provide support for:

R6 tutor/student dialogs

R7 receiving, marking, processing and returning scripts

R8 monitoring tutor performance

R9 delivering and updating of tutor notes

Challenge
Tutors are a valuable resource for the university, not only in their role as course presenters, but in that they are customer facing. By being in contact with students possibly on a day to day basis, tutors are able to collect students' comments on course materials throughout the study period. Mechanisms exist for relaying this

information back to the course authoring teams for materials improvement, but too often are such valuable perspectives lost. The availability of a fine grain interaction mechanism could facilitate their collection and use to update course materials on an on-going basis.
Derived requirements – The architecture should:
R10 provide support for fine grain interactions between tutor and course authoring team

3. An Architecture for eLT

In this section we describe in broad terms the EdJB blueprint architecture. EdJB is a proposed extension to Enterprise JavaBeans. Its goals are:

- to leverage the technology of Enterprise JavaBeans to help the Educational Engineer;
- to provide a uniform and scalable software platform upon which the development of distance learning and teaching system development can take place;
- to provide support for the business processes of distance learning and teaching.

By being based on EJB, EdJB fosters the reuse of electronic support for programmes and courses based on the development and/or acquisition of components that can be customized, assembled and deployed in a variety of settings. At the same time:

- as does EJB, it provides a standard component architecture for building distance learning and teaching electronic support, and makes the development of eLT support applications easier allowing a concentration on the business logic of learning and teaching, rather than the low-level mechanisms for security and trust, for instance;
- it inherits from EJB desirable characteristics of electronic support, such as platform independence, that make eLT available to the whole community;
- it relies upon tried and tested technologies and includes support for legacy systems so that there is no 'ground zero' for development.

The reader should note that we do not claim in this paper to develop any radical technologies; from our perspective such radicalism is not required. Besides, as EJB gains credence as the standard platform for the development and deployment of eCommerce applications, EdJB-based systems will combine seamlessly with commercial applications. Before entering into the details of the EdJB architecture, we first identify those features of the EJB architecture upon which we will build.

3.1. The EJB Architecture

The EJB architecture, described in (Shannon et al. 2000), is an application of the 3-tier architecture. As such it separates *client view* from *business logic* from *persistent storage*; moreover, interactions between the tiers are heavily (but usefully) constrained. The EJB architecture is outlined in (Fig. 1a).
One of the strengths of the EJB architecture is that it supplies *data persistence*, *security*, and *transaction management* - low-level technologies underlying many of today's eCommerce interactions. In doing this, eCommerce application development is greatly facilitated: the eCommerce developer can focus his or her efforts on capturing relevant business logic, delegating to the EJB architecture the provision of the low-level technologies. As illustrated in (Fig. 1a), EJB containers package business logic together with services. Services offered by the EJB containers are invoked through APIs (Application Programming Interfaces) defined within the architecture. Note that EJB containers and Enterprise beans, by residing in the architecture's middle-tier, are intended to be used (and usually are) for the definition of server side logic. This is perfectly adequate for current eCommerce application where most of the computation is performed by the server, with (so-called) thin clients presenting, what are usually web based, user interfaces.

3.2 The EdJB Architecture: from Requirements to Architecture

We have already noted that eCommerce and eLT share many requirements. It follows that eLT will also require data persistence, security and transaction management as basic services. We argue, however, that eLT interactions also require a new type of service that we call, for want of a better term, *cache coherence*. This requirement arises in the fine grain, but intermittent, nature of the educational dialogs (established as requirements **R5** and **R10**). Together with the reality check of Section 3, which states that we do not have a 24/7 on-line service, this implies the existence of local state information at both learner and educator sites, with needs for synchronization between them at frequent intervals. In essence, the learner site needs to act as cache to the educator's site, with synchronization to make them coherent.

As an example of the mechanism at work, consider the following likely scenario. Having registered for study on a distance learning course, Jazz downloads the first teaching component from the eLT provider's web site. A week passes in which Jazz works, off-line, through the first course component. During this period, Jazz works through 10 SAMs, 8 of which are answered correctly. In the mean time, two course material updates have been made by the eLT provider: one affects the first course component (which Jazz has), one the second (which Jazz does not yet have). At the end of the first week, and having finished the first course component, Jazz logs on to the eLT web site to download the second course component. Via this link, information is passing both ways: Jazz makes a request for the second course component; an updated version is downloaded to Jazz's machine; without knowing the details of the transaction, but with permission, Jazz's SAM performance profile is uploaded to the eLT site; updates to Jazz's copy of the first course component are made to reflect the course updates, and Jazz is informed that this has occurred.

Over and above the downloading of the second course component, looking at this dialog in terms of caches, *cached information at both sites has to be made coherent*: from Jazz to eLT provider - to synchronize Jazz's SAM profile; from eLT provider to Jazz - to synchronize course materials. This mechanism will, moreover, form a critical component of the dialog between student and provider that will be present within many, if not all, educational software development projects. Because of its omnipresence, an architectural solution cache coherence between sites should be thought of as a basic service of the architecture.

The EdJB architecture incorporates a cache coherence service, as is illustrated in (Fig. 1b). The client tier now contains local 'persistent' storage for 'caching' local information, as well as its own *EdJB container* that wraps

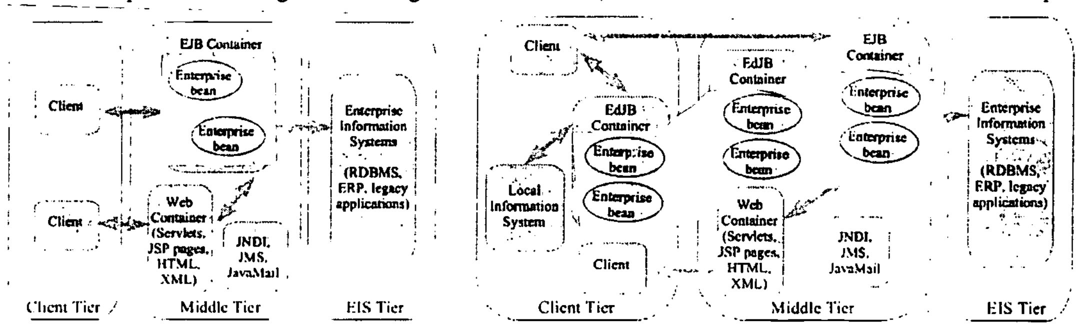

Figure 1: (a) The EJB architecture, and (b) The EdJB architecture

client business logic. This EdJB container provides cache coherence services through communication with a corresponding EdJB container in the middle-tier.

With the EdJB architecture, when off-line, the learner's interaction with the teaching materials will be through the Local Information System with the client business logic used to support it (by collecting SAM answers, for instance, etc). In off-line mode, of course, the middle and EIS tiers are not accessible for interaction. When on-line, the EdJB containers are joined so that client- and server-side data can be made coherent.

3.3. Does EdJB Satisfy the Requirements?

Requirements **R1-R10** constrain any architecture for electronic support of distance learning. We have proposed a blueprint architecture - EdJB - that we claim can provide this electronic support. The validation for the EdJB blueprint architecture against those requirements is argued in (Tab. 1).

R1, R3, R4, R6, R7	These are within the standard services provided by EJB for eCommerce
provide support for: **R2** *provide support for the use*	This is supported by the cache coherence services provided by the EdJB architecture. EdJB does not require the user to be on-line all the time, but

of changeable teaching materials as the basis of the teaching provision **R9** *the delivery and updating of tutor notes*	ensures that both local and server information systems are made consistent as often as allowed by the end-user. This provides the mechanism for supporting changeable materials: materials delivered to the student (and so residing locally) can be updated as often as the student connects to the server.
provide support for: **R5** *provide support for periodic external study validation at a fine granularity* **R10** *provide support for fine grain interactions between tutor and course authoring team*	This is supported by the cache coherence services provided by the EdJB architecture. EdJB does not require the user to be on-line all the time, while ensuring that both local and server EJB copies are consistent as often as allowed by the end-user. This provides the mechanism for supporting fine grain study validation: student 'profiles' can be generated client-side and kept coherent with server-side copies. By making these available to the tutor, again through the caching mechanism to the tutors local cache, validation of the student 'profile' can be performed and returned to the student's local cache.
R8 *provide support for monitoring tutor performance*	This is supported by the cache coherence services provided by the EdJB architecture that provides the mechanism for supporting tutor monitoring - tutor 'profiles' can be generated client-side and kept coherent with server-side copies. Verification of the tutor's performance can be performed and returned to the tutor's local cache.

Table 1: Validation of the EdJB architecture against requirements

4. Discussion and Conclusions

In this paper, we have derived an extension of the EJB architecture that provides electronic support to the distance learning experience. We have characterized the architecture, EdJB, as *cache-coherent* EJB, in that each user has local storage (the *cache*) and supporting business logic (the *coherence*) to work with off-line. The architecture provides automatic cache coherence services whenever the user goes on-line.

We have argued that the architecture meets current high-level requirements in a distance learning setting. In this section, we will reflect briefly on a reinterpretation of the distance learning challenges that motivated cache coherence in the wider context of eCommerce.

Within an eCommerce setting, product maintenance is often also in the realm of the material world. However, there are instances where product behaviour profiles are feasible and useful. This is obviously true for software products, but could also apply to a wide range of appliances with embedded electronic components – say, the controller of a modern motor car engine or even a washing machine. We can consider the profiling of product use and performance as instances of a dialog in the EdJB sense. Working within the EdJB architecture, the development of *local* product use and performance profiles with cache coherence allowing regular updating of server-side copies is facilitated, as is the downloading of software updates, the alteration of operational parameters, *etc.*, from the server-side to the local cache.

5. References

Barroca, L., Hall, J.G., Hall, P. (2000). *Software Architectures – Advances and Applications.* Springer.

Barroca, L., Hall, J.G., Hall, P. (2000), An introduction and history of software architectures, components and reuse. In (Barroca, L., Hall, J.G., Hall, P. 2000).

Bass, L., Clements, P., Kazman, R. (1998). *Software Architecture in Practice.* Addison Wesley.

Rapanotti, L., Hall, J.G. (2000), Lyceum: audio visual conferencing in distance education. *Proceedings of the 1st Annual Conference of the LTSN Centre for Information and Computer Sciences.* Edinburgh.

Rapanotti, L., Hall, J.G. (2000). Lyceum: the system and its architecture. *Proceedings of ED-ICT2000, International Conference on Information and Communication Technologies for Education.* Vienna.

Rapanotti, L., Hall, J.G. (2001). Educational JavaBeans. *The Open University, Computing Department Research Report*, no 2001/03.

Shannon, B., Hapner, M., Matena, V., Davidson, J., Pelegri-Llopart, E., Cable, L. (2000). *Java™ 2 Platform Enterprise Edition, Platform and Component Specification.* Addison-Wesley.

Shaw, M., Garlan, D. (1996). *Software Architecture: Perspectives on an emerging discipline.* Prentice Hall.

NOTICE

Reproduction Basis

☒ This document is covered by a signed "Reproduction Release (Blanket)" form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.

☐ This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").

EFF-089 (5/2002)